Answer Set Programming in a Nutshell

Thomas Eiter

Institute of Information Systems
Vienna University of Technology (TU Wien)

eiter@kr.tuwien.ac.at

GK Kolloqium “Mathematische Logik und Anwendungen”, Freiburg, Sept. 11-12, 2008

Talk Outline

Introduction

- Answer Set Programming (ASP) is a recent problem solving approach
- The term was coined by Vladimir Lifschitz [1999,2002]
- Proposed by other people at about the same time, e.g. [Marek and Truszczyński, 1999],[Niemelä, 1999]
- It has roots in KR, logic programming, and nonmonotonic reasoning
- At an abstract level, relates to SAT solving and CSP.
- Book: [Baral, 2002]

Non-Monotonic Logic Programming

- "War of Semantics" in Logic Programming (1980/90ies):
  Meaning of programs like $a \leftarrow not\ b$, $b \leftarrow not\ a$?
- Great Schism: Single model vs. multiple model semantics
- To date:
  - *Well-Founded Semantics* [van Gelder *et al.*, 1991]
  - *Stable Model Semantics (aka Answer Set Semantics)* by Gelfond & Lifschitz [1988,1991].
- Shift in LP: *Answer Sets (=models)*, not proofs, *represent solutions*!
- Need techniques *to compute models* (not proofs)
- Concerted European-level effort:

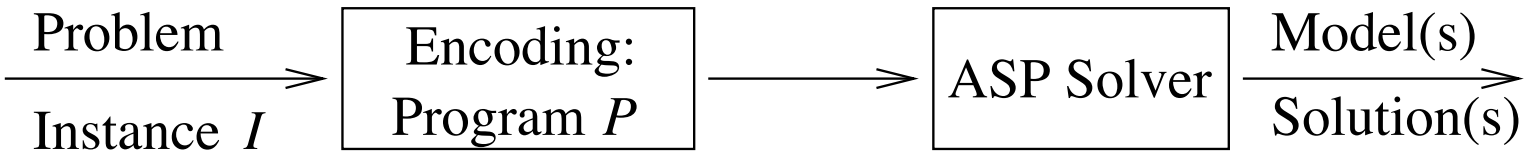
WASP

Working Group on Answer Set Programming,
IST-2001-37004, 2002-2005

ASP Paradigm

General idea: stable models are solutions!

Reduce solving a problem instance I to computing stable models of an LP



1. *Encode* I as a (non-monotonic) logic program P, such that solutions of I are represented by models of P
2. *Compute* some model M of P, using an ASP solver
3. *Extract* a solution for I from M.

Variant: Compute multiple models (for multiple / all solutions)

Nonmonotonic Logic Programs

- A *normal logic program* P is a (finite) set of rules of the form

$$a \leftarrow \ b_1, \ldots, b_m, not \ \ c_1, \ldots, not \ \ c_n$$

where all a, b_i, c_j are literals of the form p or $\neg p$, where p is a first-order atom from a (classical) FO signature.

In standard ASP, no function symbols are allowed;

  - "$\neg$" is called strong negation (also written as "$-$")
  - a may be missing (*constraint*, see later)

- In *disjunctive programs*, the rule head may be a disjunction $a_1 \vee \cdots \vee a_k$ of literals

- Notation: HB_P = set of all ground (variable-free) literals p and $-p$ with predicates and ground terms constructible from P.

Answer Sets

- The Answer Set semantics is based on 3-valued Herbrand Models (=consistent sets of ground literals $M \subseteq HB_P$), with incomplete information of the world
- For programs without "$-$," they are also called "stable models" and viewed 2-valued, with complete information of the world
- M satisfies a ground rule

$$a_1 \vee \cdots \vee a_k \leftarrow \ b_1, \ldots, b_m, not \ \ c_1, \ldots, not \ \ c_n \tag{1}$$

  if $\{b_1, \ldots, b_m\} \subseteq M$ and $M \cap \{c_1, \ldots, c_n\} = \emptyset$ implies $M \cap \{a_1, \ldots, a_k\} \neq \emptyset$.
- M satisfies ground program P, if M satisfies each $r \in P$.

Answer Sets (ctd.)

- **Key:** elimination of negation

Gelfond-Lifschitz (GL) reduct P^M

Given program P, remove from the grounding of P, $ground(P)$,

1. every rule of form (1) where some c_i is in M, and
2. all literals $not\ c_j$ from the remaining rules.

- M is an *answer set* of P iff M is a minimal model of P^M (wrt. $\subseteq$).
- Such M satisfies all rules, intuitively P justifies each atom in M.
- Note for non-disjunctive P, "a minimal" = "the least"
- Many equivalent definitions of answer set / stable model exist

Example

$$P = \{\ person(joey);$$
$$male(X) \vee female(X) \leftarrow person(X);$$
$$bachleor(X) \leftarrow male(X), not\ married(X)\ \}$$

- $M_1 = \{person(joey), male(joey), bachelor(joey)\}$ is stable
- $M_2 = \{person(joey), male(joey), married(joey)\}$ is not stable

In general, no, one, or multiple stable models exist.

Further stable model:

- $M_3 = \{person(joey), female(joey)\}$

Inconsistent Programs

- Consider the program

$$\{\ p \leftarrow not\ \ p.\ \}$$

- This program has NO answer sets
- Let P be a program and p be a new atom
- Adding

$$p \leftarrow not\ \ p.$$

to P "kills" all answer sets of P

Constraints

- Adding

$$p \leftarrow q_1, \ldots, q_m, not\ r_1, \ldots, not\ r_n, not\ p.$$

to P "kills" all answer sets of P that:

- contain $q_1, \ldots, q_m$, and
- do not contain $r_1, \ldots, r_n$

- Short:

Constraint

$$\leftarrow q_1, \ldots, q_m, not\ r_1, \ldots, not\ r_n.$$

Strong Negation

- Strong negation "$-$" is provided as possibility to express that something is provable false.
- This is different from negation as failure

Example

"At a railroad crossing, cross the rails if no train approaches."

$$walk \leftarrow at(L), crossing(L), not\ \ train\_approaches(L).$$

$$walk \leftarrow\ \ at(L), crossing(L), -train\_approaches(L).$$

- As for expressiveness "$-$" is easily compiled away; for every predicate p,
  - replace $-p$ by a fresh predicate $p_\neg$.
  - add the constraint $\leftarrow p(\vec{X}), p_\neg(\vec{X})$.

Exploiting Nondeterminism: Reviewer Selection

(1) $paper(p_1);\ paper(p_2);$
(2) $cand(\text{"}Thomas\text{"}, p_1);\ cand(\text{"}Enrico\text{"}, p_2);\ cand(\text{"}Marco\text{"}, p_2);$
(3) $assign(X, P) \leftarrow cand(X, P), not\ -assign(X, P);$
(4) $-assign(X, P) \vee -assign(Y, P) \leftarrow cand(Y, P), cand(X, P),\ X \neq Y;$
(5) $is\_assigned(P) \leftarrow assign(X, P);$
(6) $\leftarrow paper(P), not\ is\_assigned(P).$

Use disjunction in choice (3)+(4)

Answer sets:

$M_1 = \{\ldots, assign(\text{"}Thomas\text{"}, p_1), assign(\text{"}Enrico\text{"}, p_2), -assign(\text{"}Marco\text{"}, p_2)\};$

$M_2 = \{\ldots, assign(\text{"}Thomas\text{"}, p_1), assign(\text{"}Marco\text{"}, p_2), -assign(\text{"}Enrico\text{"}, p_2)\};$

Reviewer Selection: Cyclic Negation

(1) $paper(p_1);\ paper(p_2);$
(2) $cand(\text{"}Thomas\text{"}, p_1);\ cand(\text{"}Enrico\text{"}, p_2);\ cand(\text{"}Marco\text{"}, p_2);$
(3) $assign(X, P) \leftarrow cand(X, P), not\ -assign(X, P);$
(4) $-assign(Y, P) \leftarrow cand(Y, P), assign(X, P), X \neq Y;$
(5) $is\_assigned(P) \leftarrow assign(X, P);$
(6) $\leftarrow paper(P), not\ is\_assigned(P).$

(3)+(4): Choice of one element using unstratified rules (cyclic negation)

Answer sets:

$M_1 = \{\ldots, assign(\text{"}Thomas\text{"}, p_1), assign(\text{"}Enrico\text{"}, p_2), -assign(\text{"}Marco\text{"}, p_2)\};$

$M_2 = \{\ldots, assign(\text{"}Thomas\text{"}, p_1), assign(\text{"}Marco\text{"}, p_2), -assign(\text{"}Enrico\text{"}, p_2)\};$

Some Properties of Answer Sets

Minimality, Non-monotonicity

Every answer set M of P is a minimal model of P (wrt. $\subseteq$).

E.g. $P = \{a \leftarrow not\ \ b\}$: $M = \{a\}$ $P\cup = \{b \leftarrow\}$: $M = \{b\}$

Supportedness

Given an answer M of P, for every literal $a \in M$ there is some rule r from $ground(P)$ s.t. $M \models Body(r)$ and $M \cap Head(r) = \{a\}$.

But: stable $\neq$ minimal + supported! E.g. $P = \{a \leftarrow b;\ b \leftarrow a;\ a \leftarrow not\ \ a\}$

Generalization of Stratified Semantics:

If a normal P is "$-$"-free and "not " is layered ("P is stratified"), then P has a unique answer set, which coincides with the perfect model.

Failure of Cumulativity

From $a \in M$, for each answer set M of P, it does not follow that P and $P \cup \{\, a \leftarrow \ \}$ have the same answer sets (even if P has answer sets).

Computation / Complexity Issues

- For model computation, the following questions are of primary interest:

1. Computing some stable model of a given program P.
2. Computing all stable models of a given program P.
3. Given M and P, is M a stable model of P?
4. What search problems can be expressed by ASP?

- Following "classical" complexity theory, 1 and 2 have been reshaped as decision problems (does there exist some / yet another stable model)
- Problem 4 requires to resort to "nontraditional" complexity theory cf. [Marek and Remmel, 2003].
- Survey: [Dantsin *et al.*, 2001]

Basic Results

Theorem

Deciding whether a normal logic program P has some stable model is

- NP-complete in the propositional case;
- NEXPTIME-complete in the datalog (function-free) case;
- Σ_1^1-complete in the general first-order case.

- Similar results hold for deciding,
  - given P and a collection $\mathcal{C}$ of its stable models, whether P has some stable model $M \notin \mathcal{C}$;
  - given P and a ground atom a, whether $a \in M$ for some stable model M of P (brave reasoning)
- Cautious reasoning ($a \in M$ for *all* stable models of P) has complementary complexity.

Complexity: FO Logic Programs

Theorem

Deciding whether a normal logic program P with function-symbols has some stable model is Σ_1^1-complete.

Rough Sketch (for more, see [Schlipf, 1995]):

- Stable model existence of program P can be expressed by an existential second-order sentence Φ of form

$$\exists \mathbf{T} \forall \mathbf{x} \exists \mathbf{y} \phi(\mathbf{T}, \mathbf{x}, \mathbf{y}),$$

  where $\phi(\mathbf{T}, \mathbf{x}, \mathbf{y})$ is quantifier-free first order.
- Conversely, every Σ_1^1 sentence is equivalent to a sentence Φ of this form (by second-order Skolemization)
  Φ can be expressed (over the Herbrand universe) by stable model existence.

Complexity: FO Logic Programs /2

- Suppose wlog $\phi_i(\mathbf{T}\mathbf{x}, \mathbf{y}) = \bigvee_{i=1}^{n} \phi_i(\mathbf{T}, \mathbf{x}, \mathbf{y})$ is in DNF.
- Then, the program

$$
\begin{aligned}
T(\mathbf{x}) &\leftarrow not\ T'(\mathbf{x}) && \text{for each } T \in \mathbf{T} \\
T'(\mathbf{x}) &\leftarrow not\ T(\mathbf{x}) \\
sat(\mathbf{x}) &\leftarrow \phi_i^{\star}(\mathbf{T}, \mathbf{x}, \mathbf{y}) && \text{for each } i = 1, \ldots, n \\
sat(\mathbf{x}) &\leftarrow not\ sat(\mathbf{x}) \\
eq(x, x) &\leftarrow
\end{aligned}
$$

where $\phi^{\star}(\mathbf{T}, \mathbf{x}, \mathbf{y})$ is $\phi^{(}\mathbf{T}, \mathbf{x}, \mathbf{y})$ but

- "$\neg$" is replaced by "not", and
- "$=$" is replaced by "eq" (if present),

has some stable model iff Φ is true on its Herbrand universe.

Complexity of Disjunctive Stable Models

Theorem

Deciding whether a disjunctive logic program P has some stable model is

- Σ_2^p-complete in the propositional case;
- $\mathrm{NEXPTIME}^{\mathrm{NP}}$-complete in the datalog case;
- Σ_1^1-complete in the general first-order case.

- Note: no complexity increase in the full FO case (!)
- Disjunction can be compiled away:
  - Express stable model existence as $\exists \mathbf{M} \forall \mathbf{M}' \psi(\mathbf{M}, \mathbf{M}')$, where ψ is a Boolean combination of existential sentences.
  - rewrite to $\exists \mathbf{M} \forall \mathbf{x} \forall \mathbf{M}' \exists \mathbf{y} \psi'(\mathbf{M}, \mathbf{x}, \mathbf{M}', \mathbf{y})$, where ψ' is quantifier-free.
  - Emulate Arithmetic to express $\forall \mathbf{M}' \exists \mathbf{y} \psi'$ in FOL.

Expressiveness over Finite Structures

- Recall:

Fagin's Theorem

Over finite relational structures, $\Sigma_1^1 = \mathrm{NP}$

- Stable models have balanced expressiveness

Theorem

Over finite relational structures, stable model existence for

- normal logic programs $= \mathrm{NP}$ [Schlipf, 1995]
- disjunctive logic programs $= \Sigma_2^p$ [Eiter *et al.*, 1997a]

- Note: similar results hold for brave reasoning;
- For disjunctive logic programs, negation can be even restricted to equality and input relations!

ASP vs. Prolog

- **ASP features "pure" declarative programming**

  Under answer set semantics,

  - the order of program rules does not matter;
  - the order of subgoals in a rule does not matter;
  - termination is not an issue.

- **Nondeterminism in ASP:** Possibility to make guesses

- **But: function symbols are banned from standard ASP solvers**

  - Simple Horn LPs with function symbols are undecidable
  - Some decidable fragments: ω-restricted programs, finitary programs, finitely recursive programs, FDNC programs ...

    see e.g. [Baselice *et al.*, 2007], [Bonatti and Baselice, 2008], [Calimeri *et al.*, 2008] (DLV-complex), [Šimkus and Eiter, 2007]

Relationship to SAT

- Answer sets of a LP P are particular classical models of P
- Clarke aimed at characterizing the model of a (monotonic) normal program P by syntactic "completion" $Comp(P)$ [Clark, 1978]

Example

$P = \{p \leftarrow q\}$; $\quad Comp(P) = \{p \leftrightarrow q,\ q \leftrightarrow \bot\}$

- $Comp(P)$ captures the stable models of a normal LP P, if P is *tight* (= acyclic positive recursion in M).
- Does not hold in general

Example

$P = \{p \leftarrow q, q \leftarrow p\}$: $\quad Comp(P) = \{p \leftrightarrow q,\ q \leftrightarrow p\}$

$Comp(P)$ doesn't capture the single stable model $M = \{p, q\}$;

Note: P is not tight in M.

Relationship to SAT (ctd.)

- With additional *loop formulas* $LF(P)$, it is possible to capture the stable models [Lin and Zhao, 2002].

Stable models = completion + loop formulas

Intuitively, $LF(P)$ contains clauses enforcing that truth of atoms in a cycle must be supported from an atom outside.

Example (ctd.)

$P_1 = \{\ p \leftarrow q\ \}$: $LF(P_1) = \top$ $Comp(P_1) \cup LF(P_1) \equiv p \leftrightarrow q$

$P_2 = \{\ p \leftarrow q,\ q \leftarrow p\ \}$: $LF(P_2) = p \wedge q \supset \bot$ $Comp(P_2) \cup LF(P_2) \equiv \neg p \wedge \neg q$

- Also feasible for disjunctive programs + non-propositional program [Lee and Lifschitz, 2003], [Chen *et al.*, 2006].
- Based on this, some ASP solvers employ SAT solvers.
- Downside: exponentially many loop formulas in general

Relationship to SAT (ctd.)

- Benefits of ASP over SAT:
  - Transitive closure expressible
  - Express, to some extent. reasoning about multiple models
  - "high level",
  - predicates, rules, negation as failure, many constructs
- Comparisons of ASP/SAT Efficiency
  e.g., [Cadoli *et al.*, 2006], [Arieli *et al.*, 2004]
- Also relationship to / coupling with CSP has been explored
  e.g., [Baselice *et al.*, 2005]
- Some ASP systems (e.g. aspps) are strongly constraint driven.
- Constraints play an important role in model search

Relationship to Non-Classical/Non-montonic Logics

- **Nonmonotonic Logics:**
  Close relationship to other standard logics/formalisms in Nonmonotonic Reasoning (Autoepistemic Logic, Circumscription, Default Logic)

  Normal Logic Programs under Answer Set Semantics can be considered as a Fragment of Reiters Default Logic [Gelfond and Lifschitz, 1991]

- **McDermott-style Nonmonotonic Modal Logics**
  Close relationship to nonmonotonic Logic K45 [Marek and Truszczyński, 1993]

- **Equilibrium Logic** [Pearce, 2006],[Pearce and Valverde, 2006]
  Reconstruction of Answer Set Semantics in terms of the *Logic of Here and There*

Answer Set Solvers

- NP-/ Σ_2^p-completeness: Efficient computation of answer sets is not easy!
- Need to handle, for applications
  1. complex data (large data volumes)
  2. search
- Efforts to realize tractable fragments
- Many ASP solvers are available (function-free programs)

Approach:

- Logic programming and deductive database techniques (for 1.)
- SAT/Constraint Programming techniques for 2.

Different sophisticated algorithms have been developed
(like for SAT solving)

Answer Set Solvers

DLV	`http://www.dbai.tuwien.ac.at/proj/dlv/` *
Smodels	`http://www.tcs.hut.fi/Software/smodels/` **
GnT	`http://www.tcs.hut.fi/Software/gnt/`
Cmodels	`http://www.cs.utexas.edu/users/tag/cmodels/`
ASSAT	`http://assat.cs.ust.hk/`
NoMore(++)	`http://www.cs.uni-potsdam.de/~linke/nomore/`
Platypus	`http://www.cs.uni-potsdam.de/platypus/`
clasp	`http://www.cs.uni-potsdam.de/clasp/`
XASP	`http://xsb.sourceforge.net`, distributed with XSB v2.6
aspps	`http://www.cs.engr.uky.edu/ai/aspps/`
ccalc	`http://www.cs.utexas.edu/users/tag/cc/`

* + extensions, e.g. DLVEX, DLVHEX, DLV^{DB}, DLT, DLV-complex ** + Smodels_cc

- Several provide a number of extensions to the language described here.
- Answer Set Solver Implementation: see Niemelä's ICLP'04 tutorial.
- ASP Solver competition: see LPNMR 2007 conference;
- ASPARAGUS Benchmark platform `http://asparagus.cs.uni-potsdam.de/`

Architecture of ASP Solvers

Typically, a two level architecture

1. **Grounding Step:**
   Given a program P with variables, generate a (subset) of its grounding which has the same models

2. **Model Search**:
   More complicated than in SAT/CSP Solving:
   - Candidate generation (classical model)
   - model checking (stability!)
     - for SAT, model checking is in ALOGTIME
     - for normal programs, model checking is P-complete
     - for disjunctive programs, model checking is coNP-complete

Grounding Step

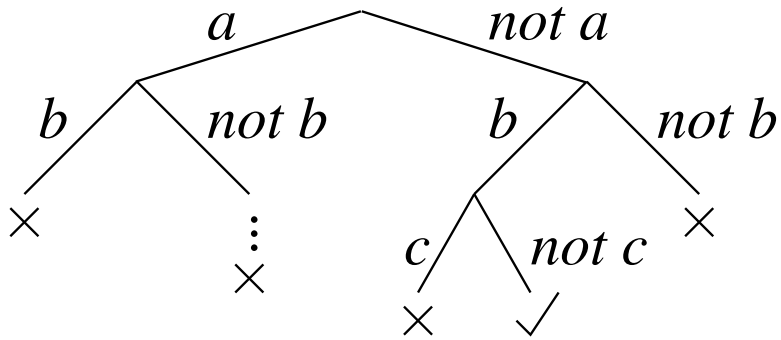
- Efficient grounding is at the heart of current systems
- Sophisticated techniques
  - DLV's grounder (built-in);
  - lparse (Smodels), gringo (clasp)
  - XASP, aspps
- Special techniques used:
  - "*Safe rules*" (DLV); every variable in a rule must occur on a unnegated atom in the body, whose predicate is not $=$ or any another built-in predicate
  - *domain-restriction* (Smodels)
- Problem: Grounding bottleneck [Eiter *et al.*, 2007]

  Research on nonground evaluation (e.g., You et al., Schaub et al.); XASP (XSB Extensions)

Model search

- Applied for ground programs.
- Different Techniques:
  - Translations to SAT (e.g. Cmodels, ASSAT)
  - tailored search procedures (Smodels, DLV, NoMore, aspps, clasp)

a :– *not* b.
b :– *not* a.
c :– *not* c, a.



  - Backtracking procedures for assigning truth value to atoms
  - Similar to DPPL algorithm for SAT
  - Important: Heuristics (which atom/rule to consider next); involved
- Stability check: unfounded sets, reductions to UNSAT

ASP Extensions

- Many extensions have been proposed, partly motivated by applications
- Some are syntactic sugar, other strictly add expressiveness
- Incomplete list:
  - nested expressions
  - cardinality constraints (Smodels)
  - optimization: weight constraints, *minimize* (Smodels); weak constraints (DLV)
  - aggregates (Smodels, DLV)
  - templates (for macros, DLT), external functions (DLVEX,DLVHEX)
  - Frame Logic syntax (for Semantic Web)
  - preferences: e.g., PLP, ordered programs
  - KR frontends (diagnosis, abduction, planning,...) in DLV
- Comprehensive survey of extensions (2005):

  `http://www.tcs.hut.fi/Research/Logic/wasp/wp3/`

Weak Constraints

- Allow the formalization of optimization problems in an easy and natural way.
- Constraints vs. weak constraints:
  - Constraints "kill" unwanted models;
  - Weak constraints express desiderata which should be satisfied, if possible.
- The answer sets of a program P with a set W of weak constraints are those answer sets of P which minimize the number of violated constraints.
- Such answer sets are called *optimal or best models of* (P, W).
- Other solvers feature similar constructs.

Guess-Check-Optimize Methodology

- Complements a natural "Guess & Check" Methodology
- Use weak constraints to filter out best (optimal) solutions
- Divide P into three main parts:

Guessing Part

$G \subseteq P$: The answer sets of $G \cup F_I$ represent "solution candidates" for instance I.

Checking Part (optional)

$C \subseteq P$: The answer sets of $G \cup C \cup F_I$ represent the admissible solutions of I.

Optimization Part (optional)

The optimization part $O \subseteq P$ consists of weak constraints, and defines an objective function $f : \mathit{Answer\_Sets}(G \cup C \cup F_I) \rightarrow \mathbb{N}$.
The answer sets minimizing f are selected.

Employee Assignment

- Goal: Divide employees in two project groups p_1 and p_2.
  1. Skills of group members should be different.
  2. Persons in the same group should not be married to each other.
  3. Members of a group should possibly know each other.
- Requirement 1) is more important than 2) and 3), which are equally important
- Layers [:x] express the relative importance of the requirements.
- ASP Solution (DLV) [Leone *et al.*, 2006]

```
assign(X,p1) v assign(X,p2) :-  employee(X).
:~ assign(X,P), assign(Y,P), same_skill(X,Y).    [:2]
:~ assign(X,P), assign(Y,P), married(X,Y).       [:1]
:~ assign(X,P), assign(Y,P), X!=Y, not know(X,Y).[:1]
```

ASP Applications

Problems in different domains, see

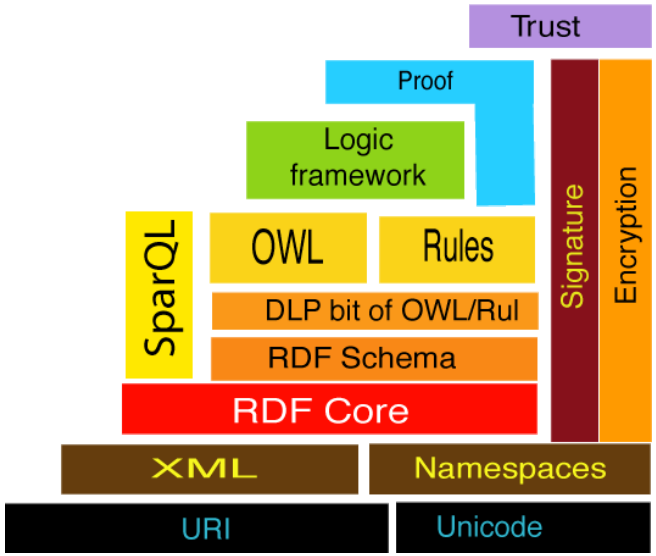
`http://www.kr.tuwien.ac.at/projects/WASP/report.html`

- information integration
- constraint satisfaction
- planning, routing
- diagnosis (e.g., Space shuttle reaction control [Nogueira *et al.*, 2001])
- security analysis
- configuration
- computer-aided verification
- biology / biomedicine
- knowledge management
- ...

ASP Front-Ends

- Some systems/groups offer *front-ends* for specific applications
- These frontends might be built-in or on top of an underlying ASP system
- DLV, e.g., offers a range of frontends
  - diagnosis
  - planning
  - inheritance reasoning
  - SQL3
  - ...
- Development of frontends for applications is a major research stream in ASP

ASP for the Semantic Web



- *RDF (Resource Description Framework)* is the SW data model
- RDF Schema enriches RDF by simple taxonomies and hierarchies
- More expressive: *OWL (Web Ontology Language)* which builds on Description Logics
- Rule languages: *Rule Interchange Format (RIF)* WG of W3C
- ASP in this context: see [Eiter, 2007]

ASP around Ontologies

- Different ways to exploit ASP and ASP techniques for ontologies:
  - **as a host language for ontology reasoning tasks**
    - Mapping / encoding of ontologies and DLs into ASP
    - Encoding of web query languages
  - **for (ad hoc) ontology management:** alignment, integration, merging, etc
  - **as a component for adding / integration rules with ontologies**
- Several extensions of ASP are geared towards this, e.g.,
  - Open Answer Set Programs [Heymans *et al.*, 2007]
  - Stable model theory for Rules on RDF/S: [Analyti *et al.*, 2005]
  - Nonmonotonic dl-programs / NLP-DL [Eiter *et al.*, 2008]
  - HEX programs /DLVHEX [Eiter *et al.*, 2005]
  - ONTODLP / ONTODLV [Calimeri *et al.*, 2003], [Ricca *et al.*, 2008]

Combining Rules and Ontologies

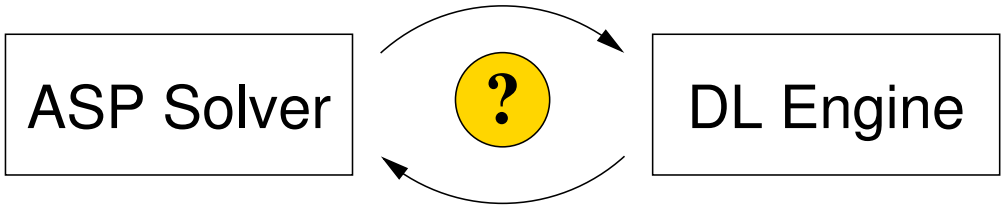
- Major Issue: Combining rules and ontologies (logic framework)
- ASP and ontology formalisms like RDF/s, OWL resp. Description Logics have related yet different underlying settings
- This makes combination non-trivial
- At the heart, the difference is between LP and Classical logic
- **Main Differences:**
  - Closed vs. Open World Assumption (CWA vs. OWA)
  - Negation as failure, strong negation vs. classical negation m
  - Unique names assumption (UNA), treatment of equality

supplier	branch	address
Barrilla	Roma	Piazza Espagna 1
Barilla	Milano	Via Cadorno 2
DeCecco	Roma	Via Salaria 10

Non-monotonic dl-Programs

- An extension of answer set programs with *queries to DL knowledge bases* (through dl-*atoms*) [Eiter *et al.*, 2008]
- dl-atoms allow to query a DL knowledge base differently

  bidirectional flow of information, with clean technical separation of DL engine and ASP solver ("loose coupling")



- Use dl-programs as "glue" for combining inferences on a DL base.
- System Prototype: NLP-DL
  `http://www.kr.tuwien.ac.at/research/systems/semweblp/`;
- ONTODLP/ONTODLV [Ricca *et al.*, 2008]: ASP + native ontologies + OWL interface

Data Integration

- Integrate data from local sources into global database, obeying local/global schema constraints [Lenzerini, 2002]
- Inconsistencies $\Rightarrow$ global database "Repairs" [Arenas *et al.*, 1999]:
- But: multiple repairs, query answering on them is quickly intractable
- Use ASP as an expressive formalism for *em repair specification and query answering* [Arenas *et al.*, 2000] etc.
- **Infomix** (IST-2002-33570; TU WIEN, U "La Sapienza", U Calabria, Rodan Systems): prototype system with ASP at the core [Leone *et al.*, 2005]
- Powerful information integration thanks ASP which similar systems (Hippo, ConQuer) could not host
  - can handle only limited schema constraints
  - have restrictions on queries (syntactic conditions)
- More discussion see [Eiter, 2005]

Workflow Management

- Execution management of orchestrated Web Services [Friedrich et al., 2008], in context of WS-DIAMOND (EU IST)

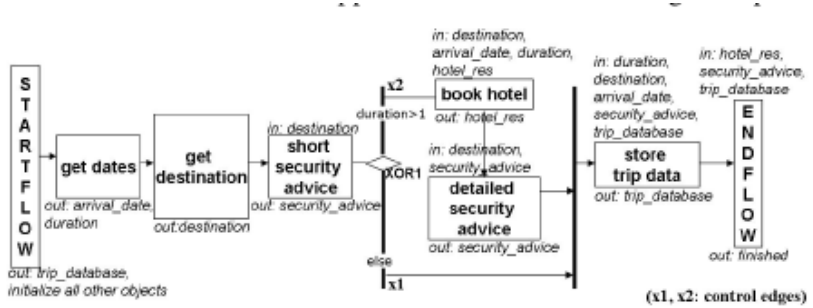


Figure 1. The trip planning example

- Problem: faulty action executions discovered; repair the workflow instance
  - **Input:** Work flow spec in WS-BPEL, exec log, faulty activities
  - **Output:** repair plan, taking contingencies into account
- Complex problem:
  - contingencies $\Rightarrow$ conditional planning
  - concurrency
  - transient errors, indeterministic actions

Workflow Management(ctd.)

- Friedrich et al. use ASP to compute repair plans
- different types of repair actions: re-execution (undo/redo), action substitution, compensation (undo)
- E.g., rules whether an activity is ok at a time point (=node in a tree):

$$
\begin{array}{ll}
-ok(A,T) & \texttt{if } failure(A,T). \\
-ok(A,T2) & \texttt{if } activity(A), -ok(A,T1), next(T1,T2), \\
 & \texttt{not } substituted(A,T2), \texttt{not } isTransFaulty(A,T1). \\
ok(A,T2) & \texttt{if } activity(A), node(T2), \texttt{not } -ok(A,T2).
\end{array}
$$

- Benefits of ASP:
  - readable specification
  - reasoning about time / time points
  - reason about dependencies (paths) in the control graph
  - inertia, default assumptions (exception handling)
- Prototype implementation in DLV: can handle medium sized orchestrated web services

Conclusion

- ASP is a recent problem sovling paradigm
- A body of theoretical work
- KR features, transitivity
- Many extensions & solvers
- Serves as a host formalism (good for prototyping)
- Assessment concerning applications:
  - N. Leone's LPNMR'07 talk

Research Topics

- Function symbols
- Program equivalence, optimization:

  E.g., *strong equivalence* [Lifschitz *et al.*, 2001] (many further notions [Woltran, 2008])

  $$P \equiv_s Q \text{ iff for all } R,\ P \cup R \text{ and } Q \cup R \text{ have the same answer sets}$$

- Proof systems (e.g., tableaux-based systems, [Gebser and Schaub, 2007], resolution [Bonatti, 2001])
- Modularity (e.g., [Janhunen *et al.*, 2007], [Eiter *et al.*, 1997b])
- Incremental model building (e.g., [Gebser *et al.*, 2008], [Calimeri *et al.*, 2007])
- Nonground ASP processing (e.g,. top down, You et al.; cf. XASP)
  Also magic sets [Faber *et al.*, 2008]), lazy grounding (e.g. Pontelli et al.)
- Debugging, software tools, methodology (e.g., [Brain *et al.*, 2007])

Expressing Disjunctive Stable Models in Σ_1^1

Rough Outline:

- Existence of a stable model is expressible by a second-order sentence

$$\exists \mathbf{M} \forall \mathbf{M}' \psi(\mathbf{M}, \mathbf{M}'), \tag{2}$$

  where $\mathbf{M}$, $\mathbf{M}'$ are lists of predicate variables and ψ is a first-order formula (in fact, a Boolean combination of existential sentences).

- This sentence can be rewritten to

$$\exists \mathbf{M} \forall \mathbf{x} \forall \mathbf{M}' \exists \mathbf{y} \psi'(\mathbf{M}, \mathbf{x}, \mathbf{M}', \mathbf{y}), \tag{3}$$

  where $\psi'(\mathbf{M}, \mathbf{x}, \mathbf{M}', \mathbf{y})$ is quantifier-free.

Lemma

Over arithmetic on the natural numbers $\mathcal{N}$ in a relational setting ($+$, $*$, exp are given by their graphs), formulas $\forall \mathbf{P} \exists \mathbf{z} \alpha(\mathbf{P}, \mathbf{z}, ...)$, where $\alpha(\mathbf{P}, \mathbf{z}, ...)$ is quantifier-free, are equivalent to first-order formulas.

Expressing Disjunctive Stable Models in Σ_1^1 (ctd.)

- We can emulate the natural numbers by a subset of the Herbrand universe, and conversely code the Herbrand universe (by Gödel numbering) to the natural numbers.
- Employing this, the formula

$$\forall \mathbf{M}' \exists \mathbf{y} \psi'(\mathbf{M}, \mathbf{x}, \mathbf{M}', \mathbf{y}) \tag{4}$$

  is (under this encoding) equivalent to a first-order formula $\alpha(\mathbf{M}, \mathbf{x})$.
- Plugging this into (3), we obtain a Σ_1^1 sentence

$$\exists \mathbf{M} \forall \mathbf{x} \alpha(\mathbf{M}, \mathbf{x}). \tag{5}$$

- The relations $+$, $*$, exp and the Gödel encoding can be provided with "guessed" predicates and first-order formulas.
- Hence, (2) is equivalent to a Σ_1^1 sentence

ASP vs Prolog: Greatest Common Divisor

Problem:

Compute the greatest common divisor of two integers $n, m > 0$

- Classical Method: Euclid's algorithm (DLV code):

$$
\begin{aligned}
gcd(X, X, X) &\leftarrow \#int(X), X > 1.\\
gcd(T, X, Y) &\leftarrow X < Y, gcd(T, X, Y1), Y = Y1 + X.\\
gcd(T, X, Y) &\leftarrow X > Y, gcd(T, X1, Y), X = X1 + Y.
\end{aligned}
$$

- Similar code in Prolog
- But: A genius like Euclid is *not* the average user

Question:

Natural encoding ?

Natural Greatest Common Divisor Encoding

ASP (DLV code):

% Declare when T divides a number N.
$divisor(T, N) \leftarrow \#int(T), \#int(N), \#int(M), N = T * M.$
% Declare common divisors

$cd(T, N1, N2) \leftarrow divisor(T, N1), divisor(T, N2).$
% Single out non-maximal common divisors T

$larger\_cd(T, N1, N2) \leftarrow cd(T, N1, N2), cd(T1, N1, N2), T < T1.$
% Apply double negation: take non non-maximal divisor

$gcd(T, N1, N2) \leftarrow cd(T, N1, N2), not\ \ larger\_cd(T, N1, N2).$

- Note: a stratified program, use of "double negation".
- A corresponding "natural" encoding in Prolog has to be more thoughtful!


Anastasia Analyti, Grigoris Antoniou, Carlos Viegar Damásio, and Gerd Wagner.
Stable model theory for extended rdf ontologies.
In *4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings*, pages 21–36, 2005.


Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki.
Consistent query answers in inconsistent databases.
In *Proc. of the 18th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'99)*, pages 68–79, 1999.


Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki.
Specifying and querying database repairs using logic programs with exceptions.
In *Proc. of the 4th Int. Conf. on Flexible Query Answering Systems (FQAS 2000)*, pages 27–41. Springer, 2000.


Ofer Arieli, Marc Denecker, Bert Van Nuffelen, and Maurice Bruynooghe.
Database repair by signed formulae.
In Dietmar Seipel and Jose Maria Turull Torres, editors, *FoIKS*, volume 2942 of *Lecture Notes in Computer Science*, pages 14–30. Springer, 2004.


Chitta Baral, Gerhard Brewka, and John S. Schlipf, editors.
Logic Programming and Nonmonotonic Reasoning, 9th International Conference, LPNMR 2007, Tempe, AZ, USA, May 15-17, 2007, Proceedings, volume 4483 of *Lecture Notes in Computer Science*. Springer, 2007.


Chitta Baral.
Knowledge Representation, Reasoning and Declarative Problem Solving.

Cambridge University Press, 2002.


Sabrina Baselice, Piero A. Bonatti, and Michael Gelfond.
A preliminary report on integrating of answer set and constraint solving.
In Marina De Vos and Alessandro Provetti, editors, *Answer Set Programming*, volume 142 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.


Sabrina Baselice, Piero A. Bonatti, and Giovanni Criscuolo.
On finitely recursive programs.
In *Proceedings 23rd International Conference on Logic Programming (ICLP 2007)*, volume 4670 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 2007.


Piero Bonatti and Sabrina Baselice.
Composing normal programs with function symbols.
In *Proceedings 24th International Conference on Logic Programming (ICLP 2008)*, Lecture Notes in Computer Science (LNCS). Springer, 2008.
To appear.


Piero A. Bonatti.
Resolution for skeptical stable model semantics.
J. Autom. Reasoning, 27(4):391–421, 2001.


Martin Brain, Martin Gebser, Jörg Pührer, Torsten Schaub, Hans Tompits, and Stefan Woltran.
Debugging asp programs by means of asp.
In Baral et al. [2007], pages 31–43.


Marco Cadoli, Toni Mancini, Davide Micaletto, and Fabio Patrizi.

Evaluating asp and commercial solvers on the csplib.
In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors, *ECAI*, pages 68–72. IOS Press, 2006.


F. Calimeri, S. Galizia, M. Ruffolo, and P. Rullo.
Ontodlp: a logic formalism for knowledge representation, 2003.
Proc. ASP 2003.


Francesco Calimeri, Susanna Cozza, and Giovambattista Ianni.
External sources of knowledge and value invention in logic programming.
Ann. Math. Artif. Intell., 50(3-4):333–361, 2007.


Francesco Calimeri, Susanna Cozza, Giovambattista Ianni, and Nicola Leone.
Computable functions in asp: Theory and implementation.
In *Proceedings 24th International Conference on Logic Programming (ICLP 2008)*, Lecture Notes in Computer Science (LNCS). Springer, 2008.
To appear.


Yin Chen, Fangzhen Lin, Yisong Wang, and Mingyi Zhang.
First-order loop formulas for normal logic programs.
In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *KR*, pages 298–307. AAAI Press, 2006.


K. Clark.
Negation as failure.
In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, 1978.


Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov.
Complexity and Expressive Power of Logic Programming.
ACM Computing Surveys, 33(3):374–425, 2001.


Thomas Eiter, Georg Gottlob, and Heikki Mannila.
Disjunctive Datalog.
ACM Transactions on Database Systems, 22(3):364–417, September 1997.


Thomas Eiter, Georg Gottlob, and Helmuth Veith.
Modular Logic Programming and Generalized Quantifiers.
In Jürgen Dix, Ulrich Furbach, and Anil Nerode, editors, *Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-97)*, number 1265 in LNCS, pages 290–309. Springer, 1997.
Extended paper CD-TR 97/108, Institut f. Informationssysteme, TU Wien 1997.


Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits.
A Uniform Integration of Higher-Order Reasoning and External Evaluations in Answer Set Programming.
In *International Joint Conference on Artificial Intelligence (IJCAI) 2005*, pages 90–96, Edinburgh, UK, August 2005.


T. Eiter, W. Faber, M. Fink, and S. Woltran.
Complexity Results for Answer Set Programming with Bounded Predicate Arities.
Annals of Mathematics and Artificial Intelligence, 51(2-4):123–165, 2007.


Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits.

Combining Answer Set Programming with Description Logics for the Semantic Web.
Artificial Intelligence, 172(12-13):1495–1539, 2008.
doi:10.1016/j.artint.2008.04.002.

Thomas Eiter.
Data integration and answer set programming.
In Chitta Baral, Gianluigi Greco, Nicola Leone, and Giorgio Terracina, editors, *Proceedings of the 8th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2005)*, number 3662 in LNCS, pages 13–25. Springer, 2005.
Invited paper.

Thomas Eiter.
Answer set programming for the semantic web (tutorial).
In Ilkka Niemelä and Veronika Dahl, editors, *Proceedings 23th International Conference on Logic Programming (ICLP 2007)*, number 4670 in Lecture Notes in Computer Science, pages 23–26. Springer, 2007.

Wolfgang Faber, Gianluigi Greco, and Nicola Leone.
Magic sets for data integration.
In Dieter Fox and Carla P. Gomes, editors, *AAAI*, pages 1528–1531. AAAI Press, 2008.

G. Friedrich et al.
Model-based repair of web service processes.
Technical Report 2008/001, ISBI research group, University of Klagenfurt, 2008.
draft.

Martin Gebser and Torsten Schaub.

Generic tableaux for answer set programming.
In Verónica Dahl and Ilkka Niemelä, editors, *ICLP*, volume 4670 of *Lecture Notes in Computer Science*, pages 119–133. Springer, 2007.


M. Gebser, B. Kaminski, M. Ostrowski B. Kaufmann, T. Schaub, and S. Thiele.
Engineering an incremental asp solver.
In *Proceedings 12th International Workshop on Nonmonotonic Reasoning (NMR-2008)*, Sep 2008.
To appear.


M. Gelfond and V. Lifschitz.
The Stable Model Semantics for Logic Programming.
In *Logic Programming: Proceedings Fifth Intl Conference and Symposium*, pages 1070–1080, Cambridge, Mass., 1988. MIT Press.


M. Gelfond and V. Lifschitz.
Classical Negation in Logic Programs and Disjunctive Databases.
New Generation Computing, 9:365–385, 1991.


Stijn Heymans, Davy Van Nieuwenborgh, and Dirk Vermeir.
Open answer set programming for the semantic web.
J. Applied Logic, 5(1):144–169, 2007.


Tomi Janhunen, Emilia Oikarinen, Hans Tompits, and Stefan Woltran.
Modularity aspects of disjunctive stable models.
In Baral et al. [2007], pages 175–187.


Joohyung Lee and Vladimir Lifschitz.

Loop Formulas for Disjunctive Logic Programs.
In *Proceedings of the Nineteenth International Conference on Logic Programming (ICLP-03)*, pages 451–465. Springer Verlag, December 2003.


Maurizio Lenzerini.
Data integration: A theoretical perspective.
In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.


Nicola Leone, Thomas Eiter, Wolfgang Faber, Michael Fink, Georg Gottlob, Gianluigi Greco, Giovambattista Ianni, Edyta Kalka, Domenico Lembo, Maurizio Lenzerini, Vincenzino Lio, Bartosz Nowicki, Riccardo Rosati, Marco Ruzzi, Witold Staniszkis, and Giorgio Terracina.
The INFOMIX system for advanced integration of incomplete and inconsistent data.
In *Proceedings ACM SIGMOD/PODS 2005 Conference, June 13-16,2005, Baltimore, Maryland*, pages 915–917. ACM, 2005.
Demo paper.


Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello.
The DLV System for Knowledge Representation and Reasoning.
ACM Transactions on Computational Logic, 7(3):499–562, 2006.
Available via `http://www.arxiv.org/ps/cs.AI/0211004`.


Vladimir Lifschitz, David Pearce, and Agustín Valverde.
Strongly equivalent logic programs.
ACM Trans. Comput. Log., 2(4):526–541, 2001.


Vladimir Lifschitz.
Answer Set Planning.
In *Proceedings of the 16th International Conference on Logic Programming (ICLP '99)*, pages 23–37. MIT Press, 1999.


Vladimir Lifschitz.
Answer Set Programming and Plan Generation.
Artificial Intelligence, 138:39–54, 2002.


Fangzhen Lin and Yuting Zhao.
ASSAT: Computing Answer Sets of a Logic Program by SAT Solvers.
In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, Edmonton, Alberta, Canada, 2002. AAAI Press / MIT Press.


Victor W. Marek and Jeffrey B. Remmel.
On the expressibility of stable logic programming.
Journal of the Theory and Practice of Logic Programming, 3:551–567, November 2003.


W. Marek and M. Truszczyński.
Nonmonotonic Logics – Context-Dependent Reasoning.
Springer, 1993.


Victor W. Marek and Mirosław Truszczyński.
Stable Models and an Alternative Logic Programming Paradigm.
In K. Apt, V. W. Marek, M. Truszczyński, and D. S. Warren, editors, *The Logic Programming Paradigm – A 25-Year Perspective*, pages 375–398. Springer Verlag, 1999.


Ilkka Niemelä.

Logic Programming with Stable Model Semantics as Constraint Programming Paradigm.
Annals of Mathematics and Artificial Intelligence, 25(3–4):241–273, 1999.


Monica Nogueira, Marcello Balduccini, Michael Gelfond, Richard Watson, and Matthew Barry.
An a prolog decision support system for the space shuttle.
In Alessandro Provetti and Tran Cao Son, editors, *Answer Set Programming*, 2001.


David Pearce and Agustín Valverde.
Quantfied equilibrium logic.
Technical report, Universidad Rey Juan Carlos, 2006.


David Pearce.
Equilibrium logic.
Ann. Math. Artif. Intell., 47(1-2):3–41, 2006.


Francesco Ricca, Lorenzo Gallucci, Roman Schindlauer, Tina Dell'armi, Giovanni Grasso, and Nicola Leone.
OntoDLV: An ASP-based System for Enterprise Ontologies.
Journal of Logic and Computation, 2008.
doi:10.1093/logcom/exn042.


J.S. Schlipf.
Complexity and Undecidability Results in Logic Programming.
Annals of Mathematics and Artificial Intelligence, 15(3/4):257–288, 1995.


Mantas Šimkus and Thomas Eiter.
FDNC: Decidable non-monotonic disjunctive logic programs with function symbols.

In N. Dershowitz and A. Voronkov, editors, *Proceedings 14th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2007)*, number 4790 in LNCS, pages 514–530. Springer, 2007.

A. van Gelder, K.A. Ross, and J.S. Schlipf.
The Well-Founded Semantics for General Logic Programs.
Journal of the ACM, 38(3):620–650, 1991.

Stefan Woltran.
A common view on strong, uniform, and other notions of equivalence in answer-set programming.
TPLP, 8(2):217–234, 2008.